Laboratorio di Algoritmi e Strutture Dati

Docente: Camillo Fiorentini

16 ottobre 2007

1 Rappresentazioni di caratteri in C

In C un *carattere* è una **costante intera** il cui valore è dato dal codice ASCII corrispondente. Un carattere va scritto fra apici semplici ('a', 'B', '2'. ...). Il simbolo '\' (carattere di *escape*) permette di definire ulteriori caratteri. Ad esempio, '\n' è il carattere *newline* (nuova linea) il cui valore è 10, '\0' è il carattere *nullo* avente valore 0.

Carattere	Valore numerico
'a'	97
'b'	98
'A'	65
'B'	66
'0'	48
'1'	49
'<'	60
' '	32
'\n'	10
'\0'	0

(si noti che ' ' è il carattere spazio). Non esiste un "tipo astratto" carattere su cui sono definite le operazioni significative per i caratteri, ma sui caratteri sono applicabili tutte le operazioni ammesse per gli interi (visione poco astratta, ma occorre tener presente che il C è un linguaggio di basso livello).

Esempio
Supponiamo di aver definito la variabile `n` di tipo `int`. Dopo l'assegnamento

```
n = 'a' + 2;
```

il valore di `n` è 99, che corrisponde al valore del carattere 'c'. È possibile eseguire l'assegnamento

```
n = 'a' * 'b' + 1000;
```

Il valore di `n` è $97 \times 98 + 1000$, ossia 10506, che non corrisponde ad alcun carattere del codice ASCII.

1.1 Lettura di un carattere da standard input

Le operazioni di lettura avvengono tramite lo *standard input (stdin)*, che è una sequenza (*stream*) di caratteri. Per default, lo standard input è associato alla tastiera.
La funzione standard `getchar()` legge un carattere da standard input e restituisce il carattere letto.

Esempio
Dopo le linee di codice

```
int c;
c = getchar();
```

la variabile `c` di tipo `int` contiene il carattere letto da standard input da `getchar()`.

1.2 Redirezione dello standard input

Normalmente lo standard input è associato alla tastiera. È possibile *redirige* lo standard input dando il comando

```
a.exe < in.txt
```

In questo modo lo standard input è associato al file `in.txt` (che deve trovarsi nella stessa directory di `a.exe`, altrimenti occorre dare il path completo del file). Questo significa che i caratteri sono letti dal file `in.txt` anziché da tastiera. Si tenga presente che la redirezione viene fatta dal sistema, non dal programma (per il programma lo standard input è una sequenza di caratteri, quindi il programma non è consapevole del fatto che l'input sia inserito da tastiera o sia letto da un file tramite redirezione dello standard input).

1.3 End-of-file

Fra i caratteri leggibili da standard input c'è anche il segnale di "end-of-file".

- Se lo standard input è associato alla tastiera il carattere "end-of-file" può essere inserito con apposite combinazioni di tasti (in genere, $< control > +d$ con Unix, $< control > +z$ con MS-DOS), da non confondere con il segnale di interruzione del programma (tipicamente, $< control > +c$).
- Se lo standard input è rediretto da un file, il carattere "end-of-file" corrisponde alla fine file.

Quando `getchar()` legge il carattere "end-of-file" restituisce il valore dell'identificatore `EOF`, definito in `stdio.h`. Tipico valore di `EOF` è -1 (non può essere un valore positivo perché si confonderebbe con i valori dei caratteri).

```
/* file stdio.h */
...
# define EOF (-1)
...
```

1.4 Scrittura di un carattere su standard output

La funzione standard `putchar()` stampa su *standard output (stdout)* il carattere passato come argomento. Per default lo standard output è associato al terminale.

Esempio

```
putchar('a');
```

stampa

```
a
```

Si ottiene lo stesso risultato con

```
putchar(97);  // 'a' vale 97
```

Ovviamente la prima notazioe è da preferire, in quanto il codice è più leggibile. Eseguendo

```
int n = 49;
putchar(n);
```

viene stampato 1. Lo stesso risultato si ottiene ponendo

```
int n = '1';  // '1' vale 49
putchar(n);
```

Non è sintatticamente corretta la chiamata

```
putchar('49');
```

in quanto '49' non è un carattere. L'istruzione

```
putchar('\n');   // 'n' vale 10
```

oppure

```
putchar(10);
```

stampa il carattere "a capo", e la prossima operazione di scrittura avverrà su una nuova linea. Invece,

```
putchar('\0');   // OPPURE:  putchar(0);
```

non stampa nulla.

1.5 Redirezione dello standard output

È possibile *redirigere* lo standard output, su un file `out.txt` dando il comando

```
a.exe > out.txt
```

L'output di `a.exe` è scritto nel file `out.txt`. Se il file `out.txt` non esiste, viene creato. Se il file `out.txt` esiste già, il suo contenuto è perso (fare attenzione!). Con

```
a.exe >> out.txt
```

se `out.txt` esiste già, il suo contenuto non è perso e l'ouptut di `a.exe` è scritto in coda ad esso.

2 Esercizi

Per svolgere gli esercizi usare il compilatore `gcc` (è utile l'opzione `-Wall` per stampare il maggior numero di warning possibile).

Esercizio 1

Analizzare il comportamento del programma `char.c` (sul sito)

```
#include <stdio.h>

int main(){
  int c;
  c = getchar(); /* lettura di un carattere da standard input */
  while(c != EOF){
    putchar(c);  /* scrittura di un carattere da standard output */
    c = getchar();
  }
  printf("\nFine input\n");
  return 0;
}
```

In particolare:

- Provare a inserire l'input da tastiera. L'input deve terminare con "end-of-file" (si noti che al termine dell'input viene stampato il messaggio "Fine input").
- Provare a redirigere lo standard input dal file `in.txt` sul sito (oppure da un qualunque altro file di testo).
- Provare a redirigere lo standard output.
- Scrivere una nuova funzione `main()` (per conservare il `main()` precedente, dargli un altro nome, esempio `main1()`) che si comporta come prima, in più prima di ogni carattere *newline* letto in input stampa una freccia `->` (due caratteri!). Provare a eseguire il programma redirigendo l'input dal file `char.c`.

In ambiente Linux, il comando di shell

```
echo $?  // OPPURE: echo $status
```

stampa il valore restituito dall'ultimo comando eseguito. Con `char.c`, se il programma termina normalmente viene stampato 0 (valore restituito da `main()`), se invece il programma è interrotto viene stampato un valore diverso.

Esercizio 2

Scrivere una funzione

```
int toSmall(int c)
```

che trasforma una lettera maiuscola (A,...,Z) nella corrispondente lettera minuscola. Più precisamente, se c è un carattere che rappresenta una lettera maiuscola restituisce la corrispondente lettera minuscola, altrimenti restituisce c stesso.

Si tenga presente che nella codifica ASCII le lettere minuscole vengono *dopo* le lettere maiuscole. Per svolgere l'esercizio non è necessario consultare la tabella dei codici ASCII.

Modificare la funzione `main()` di `char.c` nel modo seguente:

- il programma legge da standard input una sequenza di carratteri terminante con "end-of-file" e la stampa su standard output, cambiando le maiuscole in minuscole.

Esercizio 3

Scrivere una funzione

```
int isVowel(int c)
```

che restituisce 1 se c è una vocale maiuscola o minuscola dell'alfabeto italiano, 0 altrimenti. Usare la funzione `toSmall()` dell'esercizio precedente.

Modificare il `main()` di `char.c` nel modo seguente

(a). Quando viene letta una vocale, il programma stampa un carattere speciale `CHAR` definito mediante `#define`, (gli altri caratteri sono invece stampati normalmente). Ad esempio, definire `CHAR` con il carattere *, oppure con una vocale.

(b). Vengono stampati tutti i caratteri letti tranne le vocali. Si noti che è possibile utilizzare il `main()` definito in (a), dando a `CHAR` il valore opportuno.

Esercizio 4

Supponiamo di voler codificare un testo inglese usando la seguente codifica

$$a \to z, \quad b \to y, \quad c \to x, \quad d \to w, \quad \ldots \quad y \to b, \quad z \to a$$

Quindi la prima lettera dell'alfabeto è trasformata nell'ultima, la seconda nella penultima, e così via. Il codice deve conservare le lettere maiuscole, ad esempio, $A \to Z$.

Scrivere una funzione

```
int code(int c)
```

che restituisce la codifica di c se c è una lettera dell'alfabeto inglese, altrimenti restituisce c stesso.

Modificare il `main()` facendo in modo che venga stampata su standard output la codifica dei caratteri letti in input. Come è possibile decodificare il testo ottenuto?

Esercizio 5

Usiamo ora una codifica più complessa. in cui una lettera viene "ruotata" di un fattore $r \geq 0$. Ad esempio, se r è 2, si ha la seguente trasformazione:

$$a \to c, \quad b \to d, \quad c \to e, \quad d \to f, \quad \ldots \quad x \to z \quad y \to a, \quad z \to b$$

Si noti che essendo le lettere 26, si ha la stessa trasformazione per ogni valore r della forma $26 \times k + 2$, dove $k \geq 0$ (se non si ricorda quante sono le lettere dell'alfabeto, come si può fare per calcolarle nel programma?). La stessa trasformazione si applica alle lettere maiuscole, per cui nell'esempio precedente $A \to C$, $B \to D$, ecc.

(a). Scrivere una funzione

```
int rotate(int c, int r)
```

che restituisce il carattere ottenuto ruotando c di r ($r \geq 0$) se c è una lettera, altrimenti restituisce c stesso. Modificare il `main()` dell'esercizio precedente usando la nuova codifica. Si noti che usando valori di r multipli di 26 (esempio, 2600) la codifica non produce effetto.

Suggerimento. Supporre anzitutto che valga 'a'=0, 'b' = 1, 'c'=2, ..., 'a'=25. Quale operazione permette di assegnare a $c + r$ l'intero corrispondente al carattere c ruotato di r (secondo la nuova codifica)?

Per risolvere il problema di partenza, occorre "traslare" la codifica ASCII in modo da ricondursi al caso precedente.

Conviene definire il valore di rotazione mediante una macro `ROT` posta all'inizio del programma.

(b). Dato un valore di rotazione r, qual è il valore di rotazione r' da usare per la decodifica? Verificarlo generando due eseguibili a e a' corrispondenti ai valori r e r', quindi usare a per codificare un testo e a' per decodificarlo. Controllare che si sia ottenuto il testo di partenza.